



使用 **Python** Astra Automation

NetApp
December 01, 2023

目录

使用 Python	1
NetApp Astra Control Python SDK	1
原生 Python	2

使用 Python

NetApp Astra Control Python SDK

NetApp Astra Control Python SDK 是一个开源软件包，可用于自动部署 Astra Control。该软件包也是了解 Astra Control REST API 的宝贵资源，或许可以在创建您自己的自动化平台时使用。



为了简单起见，本页其余部分将 NetApp Astra Control Python SDK 称为 * SDK *。

两个相关软件工具

SDK 包含两个不同但相关的工具，这些工具在访问 Astra Control REST API 时在不同的抽象级别运行。

Astra SDK

Astra SDK 提供核心平台功能。它包括一组 Python 类，用于抽象化底层 REST API 调用。这些类支持对各种 Astra Control 资源执行管理操作，包括应用程序，备份，快照和集群。

Astra SDK是软件包的一部分、并在单个软件包中提供 `astraSDK.py` 文件您可以将此文件导入到您的环境中并直接使用这些类。



NetApp Astra Control Python SDK*（或仅 SDK）是整个软件包的名称。* Astra SDK*是指单个文件中的核心Python类 `astraSDK.py`。

工具包脚本

除了 Astra SDK 文件之外、还可以使用 `toolkit.py` 脚本也可用。此脚本可通过访问内部定义为 Python 函数的独立管理操作，在较高的抽象级别下运行。该脚本将导入 Astra SDK 并根据需要调用类。

如何访问

您可以通过以下方式访问 SDK。

Python 软件包

SDK 可从获取 "[Python 软件包索引](#)" 名称为 `actoolkit`。软件包将分配一个版本号，并将根据需要进行更新。您必须使用 * **Pip** 软件包管理实用程序将软件包安装到您的环境中。

安装后、可以通过放置来利用 `astraSDK.py` 类 `import astraSDK` 在脚本中。此外、`actoolkit` 可以直接在命令提示符处调用、相当于 `toolkit.py (actoolkit list clusters 与相同 ./toolkit.py list clusters)`。

请参见 "[PyPI：NetApp Astra Control Python SDK](#)" 有关详细信息 ...

GitHub 源代码

此外，还可以从 GitHub 获取 SDK 源代码。存储库包括以下内容：

- `astraSDK.py` (采用Python类的Astra SDK)
- `toolkit.py` (基于功能的高级脚本)

- 详细的安装要求和说明
- 安装脚本
- 其他文档

您可以克隆 "[GitHub : NetApp/NetApp-Astra-toolkits.](#)" 存储库连接到本地环境。

安装和基本要求

在安装软件包并准备使用该软件包时，需要考虑多个选项和要求。

安装选项摘要

您可以通过以下方式之一安装 SDK：

- 使用准备好的 "[Docker: NetApp/Astra-toolkits.](#)" 映像、已安装所有必要的依赖项、包括 `actoolkit`
- 使用Pip安装 `actoolkit` 从PyPI打包到Python环境
- 克隆GitHub存储库并复制/修改两个核心Python文件、以便可以通过Python客户端代码访问它们

有关详细信息，请参见 [PyPI](#) 和 [GitHub](#) 页面。

Astra Control 环境的要求

直接使用Astra SDK中的Python类还是中的功能 `toolkit.py` 脚本、最终您将在部署Astra Control时访问REST API。因此，您需要一个 Astra 帐户以及一个 API 令牌。请参见 "[开始之前](#)" 有关详细信息，请参见本文档 * 入门 * 一节中的其他页面。

NetApp Astra Control Python SDK 的要求

SDK 具有与本地 Python 环境相关的几个前提条件。例如、您必须使用Python 3.8或更高版本。此外，还需要几个 Python 软件包。有关详细信息，请参见 [GitHub](#) 存储库页面或 [PyPI](#) 软件包页面。

有用资源摘要

下面是您开始使用所需的一些资源。

- "[PyPI : NetApp Astra Control Python SDK](#)"
- "[GitHub : NetApp/NetApp-Astra-toolkits.](#)"
- "[Docker: NetApp/Astra-toolkits.](#)"

原生 Python

开始之前

Python是数据中心自动化的一种流行开发语言。在将 Python 的原生功能与多个通用软件包结合使用之前，您需要准备环境和所需的输入文件。



除了使用 Python 直接访问 Astra Control REST API 之外，NetApp 还提供了一个工具包软件包，用于抽象化 API 并消除一些复杂性。请参见 "[NetApp Astra Control Python SDK](#)" 有关详细信息 ...

准备环境

下面介绍了运行 Python 脚本的基本配置要求。

Python 3.

您需要安装最新版本的 Python 3。

其他库

必须安装 * 请求 * 和 * urllib3 * 库。您可以根据环境需要使用 pip 或其他 Python 管理工具。

网络访问

运行脚本的工作站必须能够访问网络并访问 Astra Control。使用 Astra 控制服务时、您必须连接到 Internet 并能够连接到上的服务 <https://astra.netapp.io>。

身份信息

您需要一个具有帐户标识符和 API 令牌的有效 Astra 帐户。请参见 "[获取 API 令牌](#)" 有关详细信息 ...

创建 JSON 输入文件

Python 脚本依赖于 JSON 输入文件中包含的配置信息。下面提供了示例文件。



您需要根据环境的具体情况更新这些示例。

身份信息

以下文件包含 API 令牌和 Astra 帐户。您需要使用将此文件传递到 Python 脚本 `-i` (或 `--identity`) CLI 参数。

```
{
  "api_token": "kH4CA_uVIa8q9UuPzhJaAHaG1aR7-no901DkkrVjIXk=",
  "account_id": "5131dfdf-03a4-5218-ad4b-fe84442b9786"
}
```

列出应用程序

您可以使用以下脚本列出 Astra 帐户的应用程序。



请参见 "[开始之前](#)" 所需 JSON 输入文件的示例。

```
#!/usr/bin/env python3
##-----
-----
#
```

```

# Usage: python3 list_man_apps.py -i identity_file.json
#
# (C) Copyright 2022 NetApp, Inc.
#
# This sample code is provided AS IS, with no support or warranties of
# any kind, including but not limited for warranties of merchantability
# or fitness of any kind, expressed or implied. Permission to use,
# reproduce, modify and create derivatives of the sample code is granted
# solely for the purpose of researching, designing, developing and
# testing a software application product for use with NetApp products,
# provided that the above copyright notice appears in all copies and
# that the software application product is distributed pursuant to terms
# no less restrictive than those set forth herein.
#
##-----
-----

import argparse
import json
import requests
import urllib3
import sys

# Global variables
api_token = ""
account_id = ""

def get_managed_apps():
    ''' Get and print the list of apps '''

    # Global variables
    global api_token
    global account_id

    # Create an HTTP session
    sess1 = requests.Session()

    # Suppress SSL unsigned certificate warning
    urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

    # Create URL
    url1 = "https://astra.netapp.io/accounts/" + account_id +
"/k8s/v2/apps"

    # Headers and response output
    req_headers = {}

```

```

resp_headers = {}
resp_data     = {}

# Prepare the request headers
req_headers.clear
req_headers['Authorization'] = "Bearer " + api_token
req_headers['Content-Type'] = "application/astra-app+json"
req_headers['Accept'] = "application/astra-app+json"

# Make the REST call
try:
    resp1 = sess1.request('get', url1, headers=req_headers,
allow_redirects=True, verify=False)

except requests.exceptions.ConnectionError:
    print("Connection failed")
    sys.exit(1)

# Retrieve the output
http_code = resp1.status_code
resp_headers = resp1.headers

# Print the list of apps
if resp1.ok:
    resp_data = json.loads(resp1.text)
    items = resp_data['items']
    for i in items:
        print(" ")
        print("Name: " + i['name'])
        print("ID: " + i['id'])
        print("State: " + i['state'])
    else:
        print("Failed with HTTP status code: " + str(http_code))

print(" ")

# Close the session
sess1.close()

return

def read_id_file(idf):
    ''' Read the identity file and save values '''

# Global variables
global api_token
global account_id

```

```

with open(idf) as f:
    data = json.load(f)

api_token = data['api_token']
account_id = data['account_id']

return

def main(args):
    ''' Main top level function '''

    # Global variables
    global api_token
    global account_id

    # Retrieve name of JSON input file
    identity_file = args.id_file

    # Get token and account
    read_id_file(identity_file)

    # Issue REST call
    get_managed_apps()

    return

def parseArgs():
    ''' Parse the CLI input parameters '''

    parser = argparse.ArgumentParser(description='Astra REST API -
List the apps',
                                   add_help = True)
    parser.add_argument("-i", "--identity", action="store", dest
="id_file", default=None,
                       help='(Req) Name of the identity input file',
                       required=True)

    return parser.parse_args()

if __name__ == '__main__':
    ''' Begin here '''

    # Parse input parameters
    args = parseArgs()

    # Call main function

```



```
main(args)
```

版权信息

版权所有 © 2023 NetApp, Inc.。保留所有权利。中国印刷。未经版权所有者事先书面许可，本档中受版权保护的任何部分不得以任何形式或通过任何手段（图片、电子或机械方式，包括影印、录音、录像或存储在电子检索系统中）进行复制。

从受版权保护的 NetApp 资料派生的软件受以下许可和免责声明的约束：

本软件由 NetApp 按“原样”提供，不含任何明示或暗示担保，包括但不限于适销性以及针对特定用途的适用性的隐含担保，特此声明不承担任何责任。在任何情况下，对于因使用本软件而以任何方式造成的任何直接性、间接性、偶然性、特殊性、惩罚性或后果性损失（包括但不限于购买替代商品或服务；使用、数据或利润方面的损失；或者业务中断），无论原因如何以及基于何种责任理论，无论出于合同、严格责任或侵权行为（包括疏忽或其他行为），NetApp 均不承担责任，即使已被告知存在上述损失的可能性。

NetApp 保留在不另行通知的情况下随时对本文档所述的任何产品进行更改的权利。除非 NetApp 以书面形式明确同意，否则 NetApp 不承担因使用本文档所述产品而产生的任何责任或义务。使用或购买本产品不表示获得 NetApp 的任何专利权、商标权或任何其他知识产权许可。

本手册中描述的产品可能受一项或多项美国专利、外国专利或正在申请的专利的保护。

有限权利说明：政府使用、复制或公开本文档受 DFARS 252.227-7013（2014 年 2 月）和 FAR 52.227-19（2007 年 12 月）中“技术数据权利 — 非商用”条款第 (b)(3) 条规定的限制条件的约束。

本文档中所含数据与商业产品和/或商业服务（定义见 FAR 2.101）相关，属于 NetApp, Inc. 的专有信息。根据本协议提供的所有 NetApp 技术数据和计算机软件具有商业性质，并完全由私人出资开发。美国政府对这些数据的使用权具有非排他性、全球性、受限且不可撤销的许可，该许可既不可转让，也不可再许可，但仅限在与交付数据所依据的美国政府合同有关且受合同支持的情况下使用。除本文档规定的情形外，未经 NetApp, Inc. 事先书面批准，不得使用、披露、复制、修改、操作或显示这些数据。美国政府对国防部的授权仅限于 DFARS 的第 252.227-7015(b)（2014 年 2 月）条款中明确的权利。

商标信息

NetApp、NetApp 标识和 <http://www.netapp.com/TM> 上所列的商标是 NetApp, Inc. 的商标。其他公司和产品名称可能是其各自所有者的商标。